После ухода работодателя появилось время заняться чем-нибудь для себя. Решил обновить маленький учебный проект для профилирования производительности кода в реальных условиях. Крошечная библиотека, с поддержкой многопоточности для Windows и Linux помогала оценивать эффективность прикладного кода, например пользу от замены printf на to\_chars. Замеры коротких интервалов времени осуществлялись инструкцией RDTSC (для x86) и gettimeofday на других платформах. Для выбора лучших способов было необходимо оценить как минимум точность и накладные расходы функций. Об этом и собираюсь написать.

Сперва замечания:

1. Точность программного измерения временных интервалов на обычных десктопных ОС (Windows и Linux) определить весьма сложно, поэтому я ограничусь определением дискретности возвращаемых функциями значений. Зная дискретность можно утверждать, что точность не лучше;
2. Как накладные расходы я буду рассматривать только временнЫе затраты на вызов изучаемой функции;

В интернете можно найти множество вариантов программного измерения промежутков времени, но очень мало объективных сравнений этих методов. В учебных целях я провёл замеры нескольких параметров на мой взгляд наиболее популярных методов в Windows и Linux и решил поделиться результатами.

Из тех, что невозможно прочитать в документации, измерялись следующие параметры:

1. Дискретность измерений, то есть минимальная положительная разность значений, возвращаемых измеряемой функцией. Практическая ценность дискретности в том, что её легко измерить и она позволяет пусть грубо, но оценить точность изучаемой функции (точность измерения времени не может быть выше, чем дискретность). Например, класс high\_resolution\_clock позволяет получить время в наносекундах, но с дискретностью 100ns в Windows и 20ns в Linux, вызовы этой функции с меньшим интервалом будут возвращать одинаковые значения. Кстати, этот же класс на Raspbery Pi B+ имеет дискретность и вовсе около 1,5us.
2. Длительность измерения. Выполнение любого кода требует ресурсов процессора. При профилировании скорости кода желательно оказывать минимальное воздействие на изучаемый код, поэтому важно знать сколько времени выполняется сам код профайлера. Например, получение значения Time Stamp Counter с помощью инструкции RDTSC выполняется примерно за 6.2ns. на тестовой x86 машине, а функция WinAPI GetSystemTimes на той же машине выполняется в 450 раз дольше (2,8us).